The fans united will always be connected: building a practical DTN in a football stadium

Ian Wakeman, Stephen Naicken, Jon Rimmer, Dan Chalmers, and Ciaran Fisher

University of Sussex, Brighton, BN1 9QH, UK,
ianw@sussex.ac.uk, stephenn@sussex.ac.uk, jonr@sussex.ac.uk,
d.chalmers@sussex.ac.uk and crf22@sussex.ac.uk

Abstract. Football stadia present a difficult environment for the deployment of digital services, due to their architectural design and the capacity problems from the numbers of fans. We present preliminary results from deploying an Android app building an ad hoc network amongst the attendees at matches at Brighton and Hove Albion's AMEX stadium, so as to share the available capacity and supply digital services to season ticket holders. We describe the protocol, how we engaged our users in service design so that the app was attractive to use and the problems we encountered in using Android.

1 Introduction

Successful deployments of delay tolerant networks (DTNs) in real world scenarios with real user-generated traffic, where current infrastructure networks struggle, are not common.

In this paper we describe such a deployment, and the lessons learnt.

We have, near our campus, a fairly typical football stadium with a seating capacity of 30,750. It is constructed out of reinforced concrete with a metal framed roof, situated in a valley near transport connections (road and rail). When visiting the ground on a non-match day, mobile phone signal is generally acceptable - however, in some parts (particularly the concourses and rear of stands) reception is poor or unavailable. The stadium houses the club offices, as well as hosting events such as concerts, weddings and meetings, so there is a daily occupancy. During the football season, the stadium is in full use on average once a week, bringing many more people than are there most of the time to this small area.

For these few hours the mobile phone network capacity is exceeded, resulting in intermittent connectivity for voice, text and data even where a signal can be obtained – as can be seen in figure 1. Smart phone penetration is high amongst regular attendees and anecdotally we were aware of frustration with being unable to communicate – to arrange to meet friends, to comment on social networks, to obtain updates from other concurrent matches, and to find out about transport times after the match.

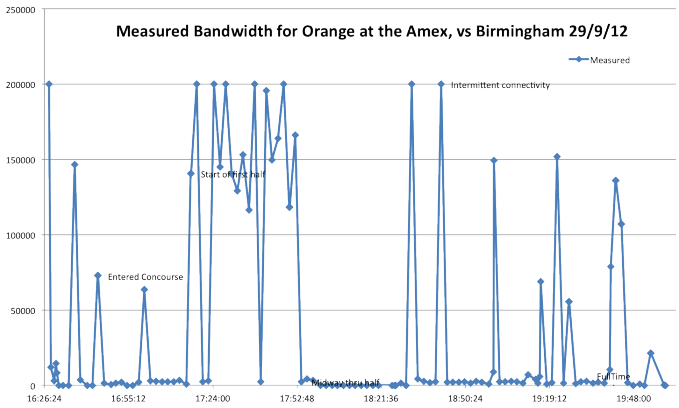

Fig. 1. Bandwidth Trace During Match To The Web, Via Mobile Phone Network, Illustrating Connectivity Problems

The financial case for improving the phone capacity for such occasional use is poor and does not solve the architectural challenges. The financial case for installing and operating an open 802.11 data network for this volume of users is also poor: the capital cost is high and the reward to the club is largely through user satisfaction. From our own experience in attending football matches, it appeared that the likely desired data services – e.g. updates from other ongoing matches, access to social networks, travel information, services supplied by the club – could be satisfied by a smartphone app providing a small set of web-based services, and that our users wouldn't desire access to the wider Internet

A delay tolerant ad-hoc network providing a distributed HTTP cache was thus a possible solution. Connectivity could be obtained indoors and in blocked areas of the stands; the few infrastructure connections could be shared so capital outlay is not required; movement of users would be sufficiently slow to allow connections to form but at half-time there would be enough movement to physically carry data out of the network dark spots in the concourse to connectivity outside; and a common interest amongst users could be used to good effect in sharing data and so improving efficiency. Our initial network simulations confirmed that the protocol seemed viable in a modelled stadium. The design of a protocol exploiting these properties is described in section 2 and the realities of implementing such a protocol on the smart phone of "the man in the street" are described in section 4.

Most users are season ticket holders, with an obvious emotional connection to the club. When we engage such users in a study, it can be assumed that they have a level of trust in the club and with their fellow fans which facilitate the sharing of resources; to be successful a DTN requires a critical mass of users to run the application and access data so engagement of users in app design, choice of services and achieving the necessary density for initial deployment required

active engagement on our part. Issues in design and deployment are described in section 3. Results, both from the network and user engagement, arising from initial deployment are described in section 5.

2 The Initial Protocol

The protocol challenge is twofold: first, to arrange for a web-view that uses this data and accommodates the delays of such a protocol without breaking the assumptions of web interactions, e.g. idempotent POST and interactivity; second (see subsection 2.3), to arrange for phones to communicate with each other to pass data to the external internet, storing and delaying forwarding where appropriate and caching to improve efficiency.

DTN protocols rely on node mobility to transfer messages. A DTN protocol assumes that the total route to the destination node is unlikely to be known or doesn't exist at a moment in time so instead a message is sent to part or all of the network and stored when a node cannot pass it on any further, termed *"Store-Carry-Forward"*. As a node moves and comes into contact with other nodes the message may be passed on until it eventually reaches its intended recipient or expires through some sort of time-out. If the standard TCP protocol was used over a highly partitioned DTN network, TCP would quickly time out due to the lack of ACKs in a short time frame so often DTN protocols work by transferring large messages node to node which would usually be encapsulated in multiple packets in TCP, an example of this is the Bundle Protocol [18].

There are a variety of modes of connectivity and nodes in the network, as illustrated in figure 2 – the key functions are outlined below and specified in detail in a patent application. Access to the network is via an app, which handles the user interaction issues, security, presentation of available data and underlying DTN connectivity.

To address security and idempotency requirements we deploy a *venue cache server* – an Internet connected server holding the cache of data for a venue. This set of data is the union of all data requested by the users and anticipated by the administrators. This will be the "origin server" in HTTP terminology. It handles app instance registration and keys for encryption, allowing user data to pass over potential malicious participant nodes without exposing data. We omit further discussion of data security and integrity here for brevity, but these issues are designed in.

2.1 GET

The app does not support general internet access, but a subset of the web – the set of valid GET requests is restricted to those URIs which are supported by the venue cache server. All GETs must be idempotent, preferably working over a uniform URI space that clearly delineates service usage. A GET may return a "bundle" of data, a set of assets either for that user or for all users. Taking as

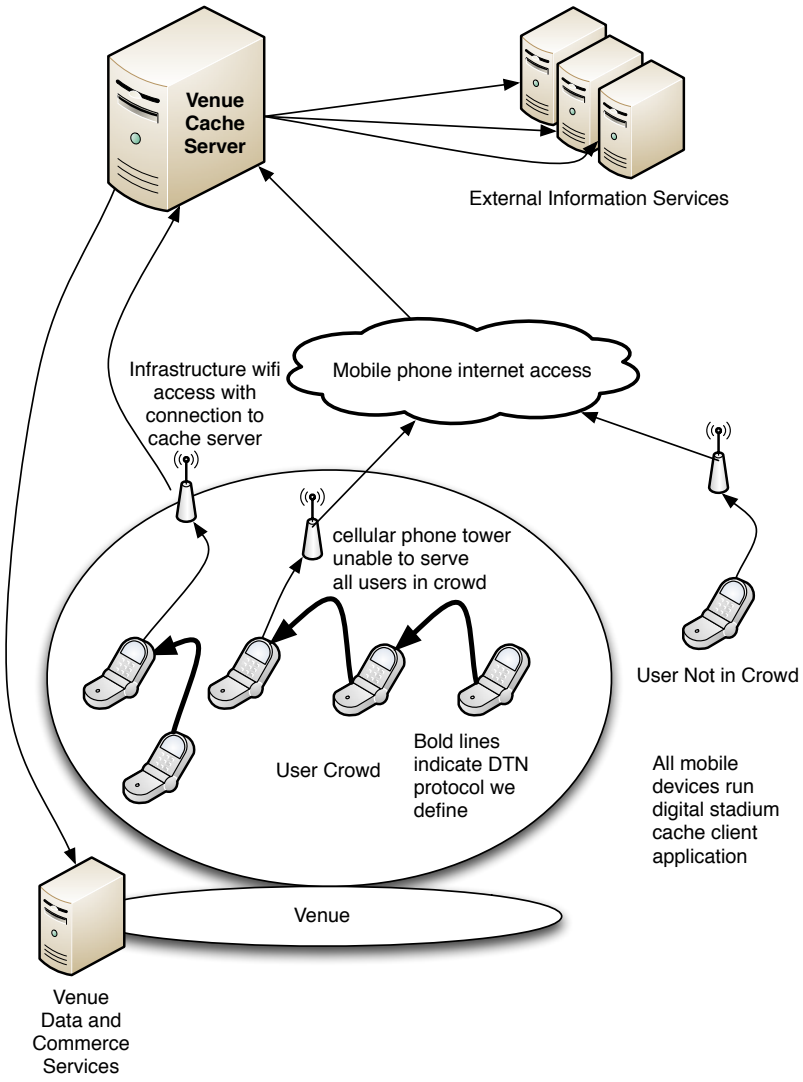

Fig. 2. System Overview

our example a "twitter" application, the following URLs would constitute the service:

`http://digital-stadium.net/club/twtr/` Provide the default up to date bundle for users who are not logged-in, e.g. a club feed.

`http://digital-stadum.net/club/twtr/u/23943` Provide the up to date bundle for the specified user, utilising their *OAuth* credentials if available.

. . . `/club/twtr/u/23943/since/8273` Provide the time-line since tweet 8273, to allow efficient updates.

Each node has a local cache. If the cache holds the data required by a GET, the display is immediate and no further communication is necessary. If there is a cache miss then the request must pass through the underlying protocol. Eventually the cache will be populated and the view updated, but in the mean time the app will allow further interaction.

2.2 POST

When the app is temporarily disconnected from a direct connection to the Internet, we would still like the app to be able to make HTTP POST calls back to the origin server. The HTTP 1.1 definition [5] makes no provision for dealing with POST commands which are broken or reset after the request is received by the origin server but generate no response. When dealing with a disconnected network, one must assume that a direct response to the POST request will not be forthcoming. We modify POST processing to work in the following fashion:

1. On receiving a POST request, the app must attempt a direct connection to the origin server. If this is possible, then the POST request executes in a normal HTTP connection.
2. If there is no direct connection to the end-point, then the app queues the POST request.
3. When a connection to another cache, including the DTN cache server, becomes available the request is forwarded.
4. Each POST request will have a unique identifier that will be included in the set of parameters encoded within the POST request, and to be included in the accompanying request headers.
5. There will be no direct response to the POST. If a response is required, then the web client must generate a GET request to a resource associated with the unique identifier.

The origin server must only process a POST request once and reject any subsequent repeat requests with the same identifier. The venue cache server will have recorded the POST request identifier in a "Confirmed POST List (see section 2.3) and thus both the request and the confirmation of its execution can follow multiple paths to / from the originating user but only having the request actioned once.

In building a service that can process POST requests, if the service requires the client side to receive a response then the service must provide a URI upon which the client can call GET, using either a direct connection or the DTN cache, to receive a valid response when the response code is 20*. If the POST call has a unique identifier of x, then the GET service may be constructed as `http://digital-stadium.net/club/response/x`

This pairing of POST/GET will allow the pairing of two AJAX calls to confirm the POST request, with the GET polling being executed using the normal

exponential back-off approach, as recommended in the HTTP specification. Issuing a request should not block interactions with other aspects of the application, or other applications, while the request is serviced. Note that the protocol below guarantees eventual execution of the POST.

2.3 Cache Maintenance Protocol

The following data lists are used to manage the protocol:

- **asset list** is the set of URLs and their last modified date that have been asked for by any of the participating nodes.
- **unconfirmed POST list (UPL)** is the list of POST requests that have been made by any of the participating nodes, including the unique identifier header and the necessary entity bodies. The UPL head is just the list of unique ids on the list.
- **confirmed POST list (CPL)** is the list of POST identifiers that are known to have been exchanged with the server.

The cache protocol works on the epidemic dissemination and updates of the asset list, the CPL and the UPL.

When a node can make a direct connection to the Internet, it will first POST the requests on the UPL, removing the items from the UPL and updating the CPL, and then enter into a *GET/if-modified* update of the URLs listed on the asset list.

When the node can make a direct connection to other participating nodes in the DTN, it will undergo the following exchanges with its peers:

1. Exchange the CPLs with each other. Merge the CPLs, and then prune any entries from the UPL as required.
2. Exchange UPL heads. Request UPL entries that are not currently on the UPL.
3. Exchange asset lists. Request assets that are new, or have a more recent last-modified date.

2.4 DTN over Android WiFi

In this section we describe the operation of the DTN cache protocol over the current Android implementation of WiFi Direct^{TM}, and other WiFi capable nodes. WiFi Direct capable phones in Android version Ice Cream Sandwich and above can act as access points for legacy systems such as those phones running Gingerbread. The venue cache serve acts as a repository for access point ids, and their associated WPA2 keys.

A node is designated *APCapable* when it can run its WiFi as an access point for other phones to connect. A node is designated in *APMode* when it is currently running as a WiFi access point. A node is designated *legacy* when it cannot run an access point, but is able to connect to an access point on another phone.

Upon starting the protocol and whenever the AP details changed, an APCapable node will upload its access point id, and its WPA2 key to the repository[1]. All participating nodes will download the set of APCapable nodes' AP ids and keys when they have a direct connection to the Internet.

Android will always attempt to use a WiFi connection in preference to a 3G/4G connection, even if the WiFi connection has no gateway to the Internet. A node will therefore only attempt to connect to the DTN on the following conditions:

1. The application is currently running in the foreground. When the application leaves the foreground, it must disconnect from the DTN.
2. If the screen or audio device is being used and the application is not at the foreground, then the node must not connect to the DTN.
3. If the screen is off and and the audio device is free, and the node is in the right temporal context (around a match time), then it should attempt to connect to the DTN.

Our aim in the protocol is to provide both spatial and temporal distribution. We thus continually reconstruct the wifi network, so that each node will connect to all of its possible neighbours in turn, thus disseminating the various request lists and assets, and adapt the number of APs to maximise coverage.

When starting to connect to the network, the APCapable node will enter APMode for a period selected from a distribution based on the $APModePeriod$ variable. We adapt this $APModePeriod$ to as to maintain a constant set of access points visible. The node will then check to see whether an infra-structure access point is available, thus allowing for the application be seeded with APs provided by the infra-structure, or if the 3G/4G connection is available. It will then connect and executed the DTNCache exchanges that are possible. On exiting APMode the node will then connect to an access point at random, for a period drawn from a legacy timer distribution. Upon termination of the connection to the access point, either due to a timeout or because the access point connection is lost due to change of mode or movement, the node will draw a new timeout from $APModePeriod$ and start the cycle again.

Legacy nodes will scan for available participating access points, and connect to a random access point for a period drawn from the legacy timer period. Upon termination of the connection to the access point, it will then attempt to make an Internet connection as described above. It will then scan for access points, and start the cycle again.

We used the ONE simulator [9] to test the efficiency of this protocol. We designed a map based upon a stand in the stadium, with a movement model which reflected the movement of people around the game - gradual arrival, movement to and from seats around the start and end of the halves, gradual dispersal - and built the protocol. The numbers demonstrated that this was a feasible

[1] Our initial tests in the protocol worked with a Galaxy Nexus which never changed its AP details. As detailed in Section 4, the need to deal with changing AP keys reduced the protocol coverage for Gingerbread devices.

approach, but we were to find out that persuading Android to implement our protocol perverted much of our original design.

3 Engaging with the Stadium and the Fans

Fig. 3. The app in action

From the project's outset the authors adopted a strong user-centric design approach to the software's development. The problems associated with, and extent to which, connectivity within the AMEX stadium were an issue for visitors were thoroughly explored. A link to an online survey was sent out to Season ticket holders as part of the Club's regular mailshot to supporters. This enabled the authors to gather fans' experiences of connectivity on match days and the kinds of technology and services they use, or would like to use, on the way to, from and at the Stadium.

1,628 fans responded and the survey reported 95% of them using smartphones and almost all reporting connectivity issues (only 6% reporting none). Just over half responded from the West Stand and this is where we targeted our recruitment efforts. We wanted to attract regular visitors to the stadium to participate in our research and we wanted to know where they would be within the stadium on match days - so ideally season ticket holders that had their own dedicated seat.

As we wanted to test the DTN within a real-world scenario and not simply the laboratory, it was vital that we provided useful services that provided users with appropriate motivation to want to use the software. We were given passes to the Stadium that granted us full access on match days and we attended every home game until the end of the season. We held a series of meetings with various Club departments (such as marketing, transport and security) in order to explore the types of information they held and the interactions they had with fans. We also held focus groups with West Stand fans before games in order to understand their match day experiences and reveal opportunities for software services development. We carried out 'accompanied journeys' with fans from their homes to the stadium, experienced all modes of organized and private transport, and constant 'participant-observation' throughout the day of each match. From these kinds of user-engagement sessions we were able to appreciate their activities, understand the variety of processes involved, the types of physical and digital artifacts used, and needs of both the stadium and the fans.

From the insights gained we began developing prototypes with the users' continual input. The key services identified that complimented the stadium experience were a version of Twitter so that they could keep up-to-date with club news and views, a live league table to show where the club was in relation to the competition in real-time, live scores at other matches, and importantly live departure boards for modes of public transport. In order for the DTN to work, fans using the developed services had to share their data connection with other service users. It was important that they were able to control shared data limits (especially as some were restricted by limits by their mobile network provider). They were not penalized for reducing their sharing and would still be able to piggyback off others.

One of the key themes emergent in our user research was the notion of *'topophilia'* (literally defined as '*love of place*'). The football stadium is often a setting that evokes strong emotions of affection and it was important that any technology we developed would not get in the way of these feelings, distract or

become a 'buzz-kill' [16]. The Club's fans are a tight-nit community with a strong sense of brand loyalty, allegiance to the team and there is a great deal of camaraderie amongst fellow fans. We too nurtured our relationship with the small group of West Stand 'beta-buddies' piloting our prototypes, maintaining regular contact, with updates, news and competitions, as well as personally meeting them at matches. It was this sense of community amongst the fans, their trust and willingness to share the app-data with each other that enhanced the success the project achieved.

4 Android 1, Designers 2

There were a number of impediments to implementing a delay tolerant networking architecture on Android. The issues were generally device-dependent and some were exasperated by poor documentation and inconsistent functionality of API calls across versions of Android. In this section, we document some of these difficulties and the solutions required to overcome them.

The delay tolerant network is implemented upon Wi-Fi Direct, which is available in Android 4.0 and above. Unfortunately, it is not possible to form P2P Wi-Fi Direct connections without requiring user intervention (Wi-Fi Protected Setup PBC or PIN) and there are also known issues in the Wi-Fi Direct group owner negotiation algorithm between Jelly Bean (4.1 - 4.2.2, JB) phones that can cause the connection to fail[2]. To avoid these issues and to allow Android Gingerbread devices (which have no Wi-Fi Direct support) to participate in the network, we make use of the `WifiManager.createGroup` API call to explicitly create a Wifi-Direct P2P group. The call to `WifiManager.createGroup` effectively creates an access point (AP) that is not tethered and can accept connections from both Wi-Fi Direct and non-Wi-Fi Direct capable devices. To distribute the configuration information so that devices can connect to an AP, two alternatives were considered, a rendezvous service and a local discovery mechanism built upon Bluetooth.

When a Wi-Fi Direct group is explicitly created, the AP configuration must be distributed to devices so that they may connect to the AP. In the existing implementation of this work, this is achieved through the use af a rendezvous service. APs upload their AP configuration details to a server over a mobile data or infrastructure Wi-Fi connection when available. Devices can connect to APs upon downloading configurations from the rendezvous and adding them to list of active configurations of their device. Clearly, this solution has obvious disadvantages in an environment where connectivity is limited. To further complicate matters, for Android versions 4.0 - 4.1.2, the AP configuration changes each time the AP is activated (i.e. the AP name and password is randomly generated each time the AP is created), therefore configurations are only valid for the lifetime of the instantiation of the AP (i.e from the time `WifiManager.createGroup` is called to the time `WifiManager.removeGroup` is called). If an AP is unable to

[2] https://code.google.com/p/android/issues/detail?id=43004

access the rendezvous service, it cannot upload its AP details and consequently will not receive connections from others. Equally, a device wishing to connect to an AP can not do so without first contacting the rendezvous.

Given the disadvantages of the rendezvous mechanism, an alternative using Bluetooth was also investigated. The proposed scheme uses insecure RFCOMM to exchange AP configurations between devices, however, insecure RFCOMM connections between certain devices may result in a pairing dialog being shown to the user after a random amount of time after the connection has been established[3]. Having replicated this issue in lab testing, it was concluded that such a prompt was undesirable and this feature was not deployed.

To resolve the bootstrapping issue, we plan to implement the following:

- Native P2P WiFi-Direct connections between Ice Cream Sandwich and Jelly Bean devices with explicit AP/group creation on Jelly Bean devices only. This should mitigate the group negotiation algorithm issue referenced above by bypassing the group negotiation algorithm.
- As it is possible to implement Wi-Fi Direct dialog interception in Android 4.2.2, only these devices become APs. Both ICS and JB clients do not show prompts to the user when connecting to a Wi-Fi Direct AP.
- As Gingerbread declines in market share, the long-term plan is to drop support for this OS.

Device dependent issues also hindered our development efforts. Wi-Fi Direct groups should remain active after the last connected device disconnects from the AP. Whilst this is the case for Nexus devices, it is not true for the Samsung S3 and possibly others. It was also observed that the Galaxy Nexus Wi-Fi driver would fail after a number of Wi-Fi Direct commands and only a reboot would restore the Wi-Fi to a working state.

5 Results

Our software was trialled in the final five home games of Brighton and Hove Albion, from April 2013. Each game was used to provide new services and to refine the protocol implementation of the app. After the initial installation, the app would automatically download a new version, which would invariably be on the morning of the match after a hard night of coding. By the final two games, our software was stable, and the DTN worked for the majority of our users. We present results below from the Wolves match, which took place on at 12:45, 4th May when the season reached a climax with the final promotion and relegation places being decided. Over the course of the match, we moved over 40 MB across the DTN. In Figure 4, we show which and how many service requests were made by our users over the course of the match. Given the changes in position of clubs over the ebb and flow of the matches, the *live scores* service was obviously most

[3] http://stackoverflow.com/questions/14804304/when-does-android-show-a-pairing-dialog-when-using-insecure-rfcomm-bluetooth-soc

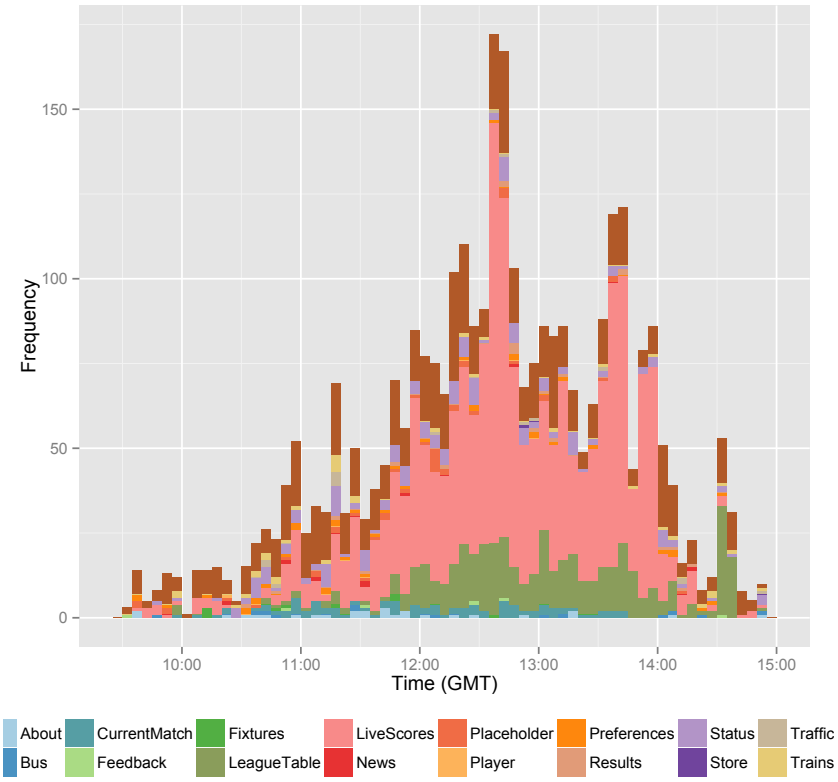

Fig. 4. What was requested when

popular. The measured round trip times for requests satisfied over 3G vs the DTN are compared in Figure 5, which show that the DTN had median rtts as a couple of minutes, which satisfied our users. We are in the process of running focus groups to refine the services and app for the coming season, but one quote shows the general level of satisfaction[4]:

> The app was a godsend on the final day of the season. It's hit-and-miss as to whether you can get a signal at the Amex, but the app meant I knew the ever-changing scores and league positions throughout the afternoon. (Simon, Patcham)

Battery consumption was a major worry with the deployment of the app. We allowed the user to configure the battery level at which the app shut down, and we had concerns that the DTN would gradually close down as the battery levels declined. Figure 6 shows the battery usage over the course of the match. The power consumed is relatively heavy, but not out of the ordinary for the use of

[4] `http://www.sussex.ac.uk/affiliates/digitalStadium/press.html`

a smartphone. Our logging was not able to disentangle how much battery was consumed by the use of the display versus the DTN, but these preliminary results show that battery consumption should not be an issue in the deployment of ad hoc networks on limited occasions.

We still had issues connecting Gingerbread devices, mostly down to bootstrapping the AP details. We are thankful that with the typical two year cycle of phone renewal in the UK, Gingerbread devices are now in a steep decline.

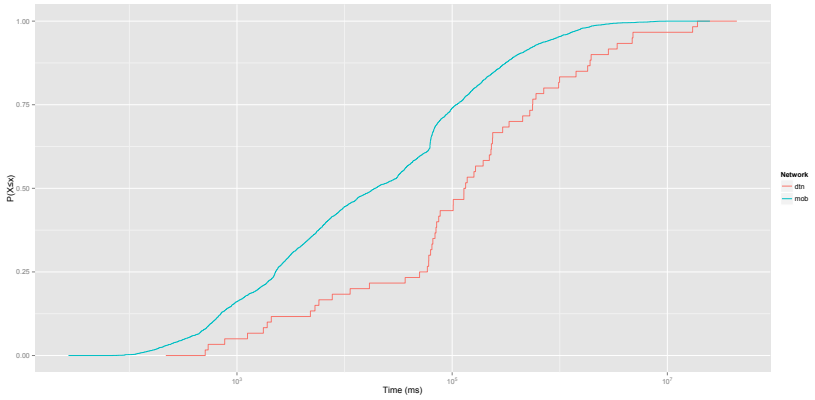

Fig. 5. Cumulative density function of round trip times for requests satisfied over 3G and over the DTN

6 Related Work

DTNs emerged as a research area in the work of Vahdat, Fall and others [4, 20] and standards such as [2]. Such protocols based on social interactions, aka "opportunistic networks" or "pocket switched networks" , where the delivery of messages is dependent upon mobile nodes coming into contact with each other, exchanging bundles of data, and then moving apart are described in e.g. [6], have been examined in the literature but rarely deployed to carry human generated traffic over standard smart phones.

Research efforts leading towards such a general purpose system include: the investigation of using personal mobile phones and PDAs with limited deployment and supporting application semantics rather than particular protocols[19], similarly Haggle [17] replaces the network stack; protocols exchanging between buses [1] and animals [8, 11] rather than human movement; a system using deployed gateway nodess [10] to occasionally connect users using a DTN protocol [18] and application gateways, which is conceptually similar to our application interface but rather different in the network and user mobility aspects; and various simulations using real-world mobility data e.g. [7].

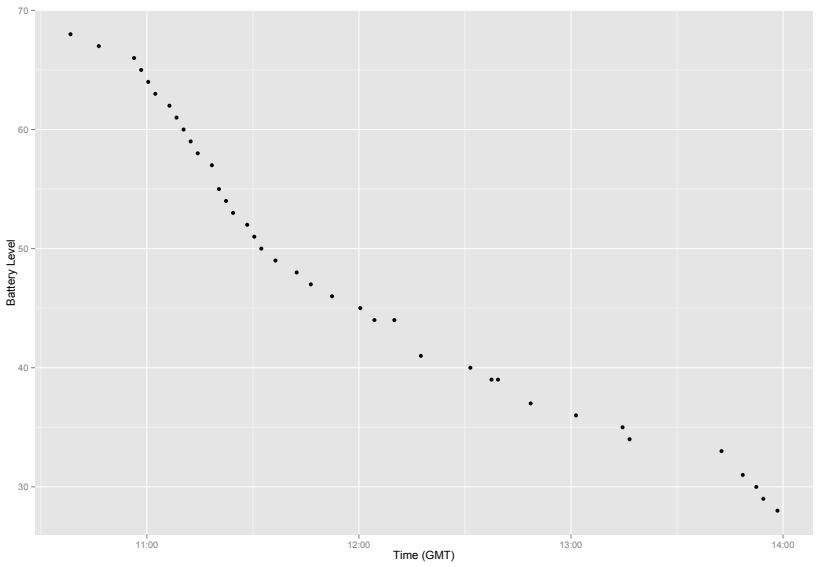

Fig. 6. Battery used over the course of the match

In recent years, opportunistic networks have been modeled as a distributed content cache [12, 15] and progress has been made in developing optimal exchange approaches [13, 3, 1]. There have been attempts by Jörg Ott and in recent internet-drafts on utilising HTTP over an opportunistic network [14].

There has been little other work on building specialised applications on smartphones for stadia. The only work of which we are aware is the estadium initiative of Edward Coyle [21], which focused on enhancing the stadium with wireless access points, and then building specific applications to work over wifi, rather than an opportunistic network between phones.

7 Conclusions

This paper describes successful initial deployments within an ongoing funded project – metaphorically we are winning, but at half time in the first game of the season. Elements of the protocol and traffic are particularly suited to this environment, but the challenges we are responding to can be found in sports and entertainment venues around the world. Development is ongoing and wider scale deployments are anticipated in the next football season, with additional services, more users and a wider range of phones.

8 Acknowledgments

This work was supported by the Engineering and Physical Sciences Research Council, grant EP/K012762/1

References

1. A. Balasubramanian, B. Levine, and A. Venkataramani. DTN routing as a resource allocation problem. *SIGCOMM Comput. Commun. Rev.*, 37(4):373–384, Aug. 2007.
2. V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, and H. Weiss. Delay-Tolerant Networking Architecture. RFC 4838, 2007.
3. V. Erramilli, M. Crovella, A. Chaintreau, and C. Diot. Delegation forwarding. In *Proc. 9th ACM intl. symposium on Mobile ad hoc networking and computing*, MobiHoc '08, pages 251–260. ACM, 2008.
4. K. Fall. A delay-tolerant network architecture for challenged internets. In *Proc. 2003 conf. on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '03, pages 27–34. ACM, 2003.
5. R. Fielding, J. Gettys, J. Mogul, H. Frystyk, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. RFC 2068 (Proposed Standard), Jan. 1997. Obsoleted by RFC 2616.
6. P. Hui, A. Chaintreau, J. Scott, R. Gass, J. Crowcroft, and C. Diot. Pocket switched networks and human mobility in conference environments. In *Proc. 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, WDTN '05, pages 244–251. ACM, 2005.
7. P. Hui, J. Crowcroft, and E. Yoneki. Bubble rap: social-based forwarding in delay tolerant networks. In *Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing*, pages 241–250. ACM, 2008.
8. P. Juang, H. Oki, Y. Wang, M. Martonosi, L. S. Peh, and D. Rubenstein. Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with zebranet. *SIGOPS Oper. Syst. Rev.*, 36(5):96–107, Oct. 2002.
9. A. Keränen, J. Ott, and T. Kärkkäinen. The ONE Simulator for DTN Protocol Evaluation. In *SIMUTools '09: Proceedings of the 2nd International Conference on Simulation Tools and Techniques*, New York, NY, USA, 2009. ICST.
10. A. Lindgren and A. Doria. Experiences from deploying a real-life DTN system. In *4th Consumer Communications and Networking Conference CCNC*, pages 217–221. IEEE, 2007.
11. T. Liu, C. M. Sadler, P. Zhang, and M. Martonosi. Implementing software on resource-constrained mobile sensors: Experiences with impala and zebranet. In *Proc. 2nd intl. conf. on Mobile systems, applications, and services*, pages 256–269. ACM, 2004.
12. A. Moghadam and H. Schulzrinne. Interest-aware content distribution protocol for mobile disruption-tolerant networks. In *IEEE intl. symp. on a World of Wireless, Mobile and Multimedia Networks Workshops (WoWMoM)*, pages 1–7, june 2009.
13. M. Musolesi and C. Mascolo. Car: Context-aware adaptive routing for delay-tolerant mobile networks. *IEEE trans. on Mobile Computing*, 8(2):246–260, 2009.
14. G. Ott and D. Kutscher. Bundling the web: Http over dtn. In *WNEPT 2006 Workshop on Networking in Public Transport*, Ontario, Canada, 2006.

15. J. Reich and A. Chaintreau. The age of impatience: optimal replication schemes for opportunistic networks. In *Proc. 5th intl. conf. on Emerging networking experiments and technologies*, CoNEXT '09, pages 85–96. ACM, 2009.
16. J. Rimmer, I. Wakeman, S. Naicken, and D. Chalmers. Digital stadium: Designing for topophilia. In *Geographic HCI 2013 Workshop*, Paris, France, April 2013. ACM CHI.
17. J. Scott, J. Crowcroft, P. Hui, C. Diot, et al. Haggle: A networking architecture designed around mobile users. In *3rd comf. on Wireless On-demand Network Systems and Services (WONS)*, pages 78–86, 2006.
18. K. Scott and S. Burleigh. Bundle Protocol Specification. RFC 5050 (Experimental), Nov. 2007.
19. J. Su, J. Scott, P. Hui, J. Crowcroft, E. De Lara, C. Diot, A. Goel, M. H. Lim, and E. Upton. Haggle:. In *Proc. 9th intl. conf. on Ubiquitous computing*, UbiComp '07, pages 391–408. Springer-Verlag, 2007.
20. A. Vahdat and D. Becker. Epidemic routing for partially-connected ad hoc networks. Technical Report CS-200006, Duke University, 2000.
21. X. Zhong, H.-H. Chan, T. Rogers, C. Rosenberg, and E. Coyle. The development and estadium testbeds for research and development of wireless services for large-scale sports venues. In *2nd intl. conf. on Testbeds and Research Infrastructures for the Development of Networks and Communities (TRIDENTCOM)*, 2006.